CSE636
Logic Programming
with Answer Sets

Slawomir Staworko
staworko@cse.buffalo.edu

Syntax
- Term = Var | Const
  \(X, Y, Z, \ldots a, b, c, \ldots\) – ground terms
- Pred = name + arity
  \(p, q, r, \ldots\)
- Atom = Pred(Term, …, Term)
  \(p(X), q(a, b, c)\) – ground atom
- Lit = Atom | ¬ Atom
  \(p(c), ¬r(a, b, c), ¬q(X, Y, c), \ldots\)
- p(a) and ¬p(a) – complementary literals

Syntax (cont)
- Rule
  \(L_1\) or … or \(L_k\) ← \(L_{k+1}\), …, \(L_m\), not \(L_{m+1}\), …, not \(L_n\).
- Fact (no body)
  \(L_1\) or … or \(L_k\) ←.
- Constraint (discussed later on)
  \(⊥\leftarrow L_1, \ldots, L_k, \text{not } L_{k+1}, \ldots, \text{not } L_n\).

Syntax, Example
\[\Pi:\]
- bird(tweety)←.
penguin(skippy)←.
doggy(puppy)←.
bird(X)←penguin(X).
fly(X)←bird(X), not penguin(X).

Semantics – Herbrand Model
- A set of literals \(S\) – Herbrand model of rule:
  \(L_1\) or … or \(L_k\) ← \(L_{k+1}\), …, \(L_m\), not \(L_{m+1}\), …, not \(L_n\).
- If \(\{L_{m+1}, \ldots, L_n\} \subseteq S\) and \(\{L_{m+1}, \ldots, L_n\} \cap S = \emptyset\)
  then \((L_{m+1}, \ldots, L_n) \cap S = \emptyset\).
- If \(S\) contains pair of complementary literals
  then \(S = \emptyset\).

Convention: if \(\Pi\) is free of classical negation (¬)
we consider only negation-free Herbrand models.

We extend Herbrand model to set of rules \(\Pi\).

Herbrand Models, Examples
\[\Pi:\]
- bird(tweety)←.
penguin(skippy)←.
doggy(puppy)←.
fly(X)←bird(X), not penguin(X).

This is Herbrand model of \(\Pi\):
\[S = \{\text{bird(tweety)}, \text{penguin(skippy)}, \text{doggy(puppy)}\}\]

\(\Pi\):
- train ← lights.
cross ← not train.
cross ← ¬train.
bird(X) ← penguin(X).
fly(X) ← bird(X), not penguin(X).

This one is not a model of \(\Pi\):
\[S = \{\text{bird(tweety)}\}\]
Semantics for positive programs

- Programs without negation-as-failure
  \[ L_1 \lor \ldots \lor L_m \leftarrow L_{m+1}, \ldots, L_n \]
- Answer sets – minimal Herbrand model
  (minimal – no subset that is a Herbrand model).
  \[ M(\Pi) \] – all answer sets of \( \Pi \)
- \( L \) is always a model, but not always a minimal one!

Answer sets – minimal Herbrand model
(minimal – no subset that is a Herbrand model).

\[ M(\Pi) \] – all answer sets of \( \Pi \)

\[ \neg p \]
\[ \neg p \]
\[ \neg p \]
\[ \neg p \]

\[ \Pi \] – logic program

We receive program \( \Pi \) from \( \Pi' \) by deleting:
- each rule with \( \neg L \) for \( L \in S \)
- \( \neg L \) from bodies of remaining rules

We continue notion
\[ M(\Pi) = \text{all answer sets of } \Pi \]

Answer set of \( \Pi \) is still a minimal Herbrand model

If \( \Pi \) is stratified then \( \Pi \) has only one model \( S \)
and \( S \) is also model of \( \Pi' \)

Semantics – Examples

Consider again \( \Pi' \):

\[ p \leftarrow \neg p \]

\( \{p\} \) is a Herbrand model of \( \Pi' \), but
\[ M(\Pi') = \emptyset \]
\[ \{\neg p\} \] is not answer set of \( \Pi' \)

Test whether \( S = \{p\} \) is an answer set
\[ M(\Pi') = \emptyset \]
So \( \{p\} \) is an answer set of \( \Pi' \)

What about \( S = \{q\} \)?
\[ M(\Pi') = \emptyset \]
So \( \{q\} \) is not an answer set of \( \Pi' \).

Answer set semantics follows intuitions

What about negation-as-failure

- Consider the following program
  \[ \Pi : p \leftarrow \neg q \]
- What is that the user wants to express?
  Fact \( p \) holds when fact \( q \) doesn’t hold.
  Since fact \( q \) is not given implicitly this should evaluate to only one solution \( \{p\} \)
- Minimal Herbrand models however are:
  \[ \{p\}, \{q\} \] – not intuitive!

Gelfond-Lifschitz reduct \( \Pi' \)

\[ S \] – set of literals, \( \Pi \) – logic program

We receive program \( \Pi' \) from \( \Pi \) by deleting:
- each rule with \( \neg L \) for \( L \in S \)
- \( \neg L \) from bodies of remaining rules

No contraposition: \( \neg p \Rightarrow q \) not equivalent to \( \neg q \Rightarrow p \)

Semantics – programs with not

- Gelfond-Lifschitz reduct \( \Pi' \)
  \[ S \] – set of literals, \( \Pi \) – logic program

We receive program \( \Pi' \) from \( \Pi \) by deleting:
- each rule with \( \neg L \) for \( L \in S \)
- \( \neg L \) from bodies of remaining rules

We continue notion
\[ M(\Pi') = \text{all answer sets of } \Pi' \]

Answer set of \( \Pi \) is still a minimal Herbrand model

If \( \Pi \) is stratified then \( \Pi \) has only one model \( S \)
and \( S \) is also model of \( \Pi' \)

Semantics – More Examples

Consider again
\[ \Pi : p \leftarrow \neg q \]
\( \{p\}, \{q\} \) – Herbrand models of \( \Pi' \)

\( \emptyset \) is only model of \( \Pi' \)
\( \emptyset \) also answer set of \( \Pi' \)
\( \emptyset \) is answer set of \( \Pi' \)

\[ \Pi' : p \leftarrow \neg q \]
\( \{p\}, \{q\} \) – Herbrand models of \( \Pi' \)
\( \emptyset \) is answer set of \( \Pi' \)

\[ \Pi : q \leftarrow q \leftarrow \neg p \]
\( \emptyset \) is answer set of \( \Pi' \)

No answer set.
Querying under Closed World Assumption
(for programs without classical negation ¬)

- Queries Q := p / ¬Q / Q ∨ Q / Q ∧ Q
- Query entailment in an answer set S:
  - S |= p if p ∈ S
  - S |= ¬Q if not S |= Q
  - S |= Q1 ∨ Q2 if S |= Q1 or S |= Q2
  - S |= Q1 ∧ Q2 if S |= Q1 and S |= Q2
- Querying a program Π
  Π |= Q if ∀S ∈ M(Π), S |= Q

Queries under CWA, Examples

Π:
1. bird(tweety) ←.
2. penguin(skippy) ←.
3. doggy(puppy) ←.
4. bird(X) ← penguin(X).
5. fly(X) ← bird(X), not penguin(X).

CWA
π |= bird(tweety)
π |= fly(tweety)
π |= penguin(skippy)
π |= ¬fly(puppy).

Queries under Open World Assumption
(for all programs)

- S |= Q (Q is true in S)
  - S |= p if p ∈ S
  - S |= ¬Q if S |= Q
  - S |= Q1 ∨ Q2 if S |= Q1 or S |= Q2
  - S |= Q1 ∧ Q2 if S |= Q1 and S |= Q2
- S |= Q (Q is false in S)
  - ¬π |= p if ¬p ∈ S
  - ¬π |= ¬Q if S |= Q
  - ¬π |= Q1 ∨ Q2 if S |= Q1 or S |= Q2
  - ¬π |= Q1 ∧ Q2 if S |= Q1 and S |= Q2

π |= Q if ∀S ∈ M(Π), S |= Q

Querying: CWA vs. OWA

Π:
1. bird(tweety) ←.
2. penguin(skippy) ←.
3. doggy(puppy) ←.
4. bird(X) ← penguin(X).
5. fly(X) ← bird(X), not penguin(X).

CWA
π |= fly(tweety)
π |= ¬fly(puppy)
π |= ¬fly(puppy).

OWA
π |= fly(tweety)
but not π |= ¬fly(puppy)
π |= ¬fly(puppy).

Integrity constraints

L ← L₁, …, Lₘ, not Lₘ₊₁, …, not Lₙ.
Remove from consideration any model S such that:
1. {L₁, …, Lₘ} ⊆ S
2. {Lₘ₊₁, …, Lₙ} ∩ S = Ø

Can easily be simulated with the following construct:
p ← L₁, …, Lₘ, not Lₘ₊₁, …, not Lₙ, not p.
Integrity constraints, Example

\[ \Pi: \]
- \( a \lor b \lor c \leftarrow \)
- \( \bot \leftarrow \neg a, \neg b, \neg p_1 \)
- \( \bot \leftarrow \neg a, \neg c, \neg p_2 \)
- \( \bot \leftarrow \neg b, \neg c, \neg p_3 \)

\[ \Pi': \]
- \( a \lor b \lor c \leftarrow \)
- \( p_1 \leftarrow \neg a, \neg b, \neg p_1 \)
- \( p_2 \leftarrow \neg a, \neg b, \neg p_2 \)
- \( p_3 \leftarrow \neg a, \neg b, \neg p_3 \)

\[ M(\Pi) = M(\Pi') = \{ \{a\}, \{b\}, \{c\} \} \]

Eliminating classical negation

- Take any program \( \Pi \) with classical negation
- Receive positive program \( +\Pi \) by substituting each rule in \( \Pi \)
- \( L_1 \lor \ldots \lor L_k \leftarrow L_{k+1}, \ldots, L_m, \neg L_{m+1}, \ldots, \neg L_n \)
- with a positive rule
- \( +L_1 \lor \ldots \lor +L_k \leftarrow +L_{k+1}, \ldots, +L_m, \neg +L_{m+1}, \ldots, \neg +L_n \)

\(-\neg p = p' \)

- Additionally for every predicate \( p \) add the following constraint:
- \( \bot \leftarrow p, p' \)

Closed FOL Queries

- First Order Logic Query \( \varphi \):
- \( \varphi = \psi \lor \varphi \lor \psi \lor \varphi \)
- \( \varphi = \exists x. \psi \varphi(x) \)
- \( \varphi = \forall x. \psi \varphi(x) \)

\( \varphi \) holds in \( \Pi \) if \( \varphi \) holds in every answer set of \( \Pi \).

SAT

- SAT problem:
  - given set of Boolean variables \( V = \{ p_1, \ldots, p_n \} \)
  - Boolean formula \( \varphi : \equiv \varphi \lor \varphi \lor \varphi \lor \varphi \)
  - is there a satisfying valuation \( \theta : V \rightarrow \{ \text{true}, \text{false} \} \), i.e.
  - \( \theta(\varphi) \neq \text{true} \)

- SAT problem is NP-complete

  We doubt there exists a polynomial program solving this problem

Eliminating classical negation, Examples

\[ \Pi: \]
1. \( \text{bird(tweety)} \leftarrow \)
2. \( \text{doggy(puppy)} \leftarrow \)
3. \( \neg \text{fly}(X) \leftarrow \text{doggy}(X) \)
4. \( \text{fly}(X) \leftarrow \text{bird}(X), \neg \text{penguin}(X) \)

\[ +\Pi: \]
1. \( \text{bird(tweety)} \leftarrow \)
2. \( \text{doggy(puppy)} \leftarrow \)
3. \( \neg \text{fly}(X) \leftarrow \text{doggy}(X) \)
4. \( \bot \leftarrow \text{fly}(X), \neg \text{fly}(X) \)
5. \( \text{fly}(X) \leftarrow \text{bird}(X), \neg \text{penguin}(X) \)

FOL Queries, Example

\( \forall x \in P. \exists y,z. \text{R}(x,y,z) \)

\[ \{ \text{fly}(X), \neg \text{fly}(X) \leftarrow \text{doggy}(X) \} \]

\( \{ \text{fly}(X), \neg \text{fly}(X) \leftarrow \text{bird}(X), \neg \text{penguin}(X) \} \)
SAT, CNF, and Example

CNF – Conjunctive Normal Form – conjunction of disjunctions of variables and negated variables

\[ \varphi = (p_1 \lor p_2 \lor q_1) \land (\neg q_1 \lor q_2) \land (\neg p_2 \lor \neg q_2 \lor \neg p_1) \]

Every formula has an equivalent formula in CNF

One of satisfying valuations for \( \varphi \) is

\[
\begin{align*}
p_1 & \rightarrow \text{false} \\
p_2 & \rightarrow \text{true} \\
q_1 & \rightarrow \text{false} \\
q_2 & \rightarrow \text{true}
\end{align*}
\]

SAT and Logic Programming

\[
\varphi = \psi_1 \land \ldots \land \psi_k
\]

\[
\psi_j = p_{j1} \lor \ldots \lor p_{jk}
\]

\[ \Pi(\varphi) : \]

for each variable \( p_i \) add rules

\[
p_i \text{ or } \neg p_i \leftarrow \bot
\]

\[
\bot \leftarrow p_i, \neg p_i
\]

\[
\bot \leftarrow \neg p_i, \neg \neg p_i
\]

for each \( \psi_j \) add rules

\[
c_j \leftarrow p_{j1}
\]

\[
\ldots
\]

\[
c_j \leftarrow p_{jk}
\]

\[
\bot \leftarrow \neg c_i
\]

\[
\bot \leftarrow \neg \neg c_i
\]

\[ \varphi \text{ has a satisfying valuation } \iff \Pi(\varphi) \text{ has an answer set.} \]

Reasoning with Exceptions

\[
\Pi : 
\begin{align*}
\text{fly}(X) & \leftarrow \text{bird}(X), \neg \text{exceptional_bird}(X) \\
\text{exceptional_bird}(X) & \leftarrow \text{penguin}(X) \\
\text{bird}(X) & \leftarrow \text{penguin}(X)
\end{align*}
\]

\[ \text{bird}(\text{tweety}) \leftarrow \\
\text{penguin}(\text{skippy}) \leftarrow
\]

\[ \Pi \models \text{fly}(\text{tweety}) \text{ and } \Pi \models \text{bird}(\text{tweety})\]  

\[ \Pi \models \text{bird}(\text{skippy}) \text{ but } \Pi \models \neg \text{fly}(\text{skippy}) \text{ [under CWA]} \]

Consistent Query Answers

Buffalo (Name, Street)
ErieCounty (Name, Street, Town)

ErieCounty: Name \rightarrow \{Street, Town\}
Buffalo (Name, Street) \subseteq ErieCounty (Name, Street, BUF)

Buffalo = \{ (John Smith, Main) \}
ErieCounty = \{ (John Smith, Elmwood, BUF) \}

There are two repairs:
1. Buffalo = \emptyset, ErieCounty = \{ (John Smith, Elmwood, BUF) \}
2. Buffalo = \{ (John Smith, Main) \}
   ErieCounty = \{ (John Smith, Main, BUF) \}

Logic Programming with Exceptions for Consistent Query Answers (cont)

3. Stabilizing rules:
   \[
   \neg \text{erie}_i(X,Y,Z) \leftrightarrow \text{erie}_i(X,S,T), \text{not (S==Y, T==Z)}
   \]
   \[
   \text{erie}_i(X,Y,\text{buf}) \leftrightarrow \text{buf}_i(X,Y)
   \]

4. Persistence rules (obsolete Copying)
   \[
   \text{buf}_i(X,Y) \leftrightarrow \text{buf}_i(X,Y), \text{not } \neg \text{buf}_i(X,Y)
   \]
   \[
   \neg \text{buf}_i(X,Y) \leftrightarrow \neg \text{erie}_i(X,Y,\text{buf})
   \]

5. Querying: (like FOL queries but in terms of